Coronavirus Classification Using Temporal Convolution Networks

Joanna Song
Department of Bioengineering
Stanford University
`songjj@stanford.edu`

Abstract

This project uses a temporal convolution network architecture to classify coronavirus genome samples into one of the 7 species that infect humans. Temporal convolution networks are made of repeating temporal residual blocks that use dilated convolutions to learn longer range dependencies. The model developed in this project explored optimal model parameters and training hyperparameters that resulted in a macro F1-Score of 0.9917 and a micro F1-Score of 0.9935 on the test set, despite training on a heavily imbalanced data set. Accurate classification of minority classes is likely due to the weighted loss function in which weights are factor in the number of class examples and a virus virulence. Lastly, grad-CAM and supervised dimension reduction via UMAP were used in an attempt to better understand model attenuation and interpret the model. This was a solo project in which I wrote the majority of code and gathered publically available data from the NCBI.

Introduction

Coronaviruses belong to a family of RNA viruses that cause disease in birds and mammals. In humans, there are 7 identified strains in total, namely HCoV-OC43, HCoV-HKU1, HCoV-229E, HCoV-NL63, MERS-CoV, SARS-CoV-1, SARS-CoV-2. The former four are endemic in populations, typically produce mild symtpoms, and account for 15%-30% of cases for the common cold every year. The latter three, however, have been responsible for multiple epidemics, including the Middle East Respiratory Syndrome outbreaks (MERS) in South Korea (2015) and Saudi Arabia (2018) as well as the Severe Acute Respiratory Syndrome (SARS) outbreak in China (2003), and one ongoing pandemic, COVID-19 (2020), in the past twenty years. MERS and SARS each boast high death rates, nearly 40% and 10%, respectively, of those infected [1]. Moreover, COVID-19 this year has already claimed over one million lives worldwide [2]. Therefore, it is crucial to study the discrepancy in virulence among strains that infect humans and develop technologies to distinguish between them.

One reliable method of coronavirus identification is Reverse Transcription Polymerase Chain Reaction (RT-PCR) of swabs from infected patients, often of nasal passages. This technology enables coronaviruses' RNA to be converted to DNA (henceforth referred as viral DNA), amplified, and detected. Therefore, it is most logical to use DNA samples when developing coronavirus classification tools, and this project will focus on applying multiple deep learning frameworks to this type of data to classify coronavirus sequences into one of 7 classes.

While there have been a few deep learning models developed for coronavirus classification, none have utilized approaches to preserve the variability of sequence lengths nor have investigated their models' interpretability. This project attempts to address both. Furthermore, it tests the Temporal Convolution Network (TCN) [3], a variation of Convolutional Neural Networks (CNN) better suited for sequential data, and compares it to LSTMs, which have also not been used for this task. These features comprise the novel contributions of this project towards coronavirus classification.

Related Work

Coronavirus Classification Models. Recently, two groups have developed models for tasks related to coronavirus classification. Habib et al. (2020) [4] developed a multi-layer perceptron classifier to sort sequences into Alpha coronavirus (HCoV-229E or HCoV-NL63), Beta coronavirus (HCoV-OC43 or HCoV-HKU1), MERS, SARS-CoV-1, SARS-CoV-2, and bronchitis-CoV. Lopez-Rincon et al. (2020) [5] uses a 3-layer convolutional neural network to perform binary classification on whether coronavirus sequences were SARS-CoV-2 strains or not. However, I believe the tasks and constraints for this project are harder to address than those in Habib et al. (2020) and in Lopez-Rincon et al. (2020). Specifically, the task at hand involves more classes that are less distinguishable. For instance, it is likely sequences from HCoV-OC43 and HCoV-HKU1 to be more conserved as both are beta coronaviruses, so differentiating between these classes is harder than between the more general subgroups of alpha and beta coronaviruses. Furthermore, Habib et al. (2020) do not limit their data to samples taken only from human hosts (i.e. they likely contained a substantial number of SARS-CoV-1 samples from other animals, such as bats), which allows their model to train on much more data and achieve better performance. Therefore, while both models serve as good references, I do not expect my models to outperform them.

Recurrent Models. Given the sequential nature of DNA sequences, recurrent models are the most intuitive choice. The classic architecture is the Recurrent Neural Network (RNN), but it is susceptible to vanishing and exploding gradients as well as a decreased ability to store long-term dependencies. Long Short Term Memory (LSTM) attempt to fix these problems with cells that contain 3 gates: an input, output, and forget gate which controls newly calculated, residual, and older calculated information through the model. Because of these benefits, a vanilla LSTM was chosen as the baseline model at the start of the project.

Convolutional Neural Networks. One major setback of recurrent models are longer training times compared to other architectures, like convolutional neural networks. Therefore, I found it worthwhile to investigate these alternatives.

Character-level convolution networks seemed most relevant for nucleotide data, as processing each base pair separately is more reasonable than trying to segment sequences (i.e. produce "words") in ways that made sense biologically. Hence, Kim et al. [6] and Zhang et al. [7] were good initial starting points for how to incorporate CNNs for additional feature extraction, and I originally planned to develop a convolutional LSTM model, similar to Shi et al. [8]. I envisioned first using convolutional layers as feature extractors to transformed the one-hot vector inputs into embeddings. These would then be concatenated, to preserve some information of relative positioning, and fed into the LSTM for sequential processing.

However, Bai et al. [3] developed a Temporal Convolutional Network (TCN) that does not use any recurrent layers but still manages to capture long-range dependencies in sequence-type data. As the name suggests, TCNs are mainly made of many temporal blocks, shown in Figure 1. These blocks repeats twice the structure of a dilated convolutional layer followed by weight normalization, a ReLU activation, and dropout, before adding it to the input which creates a residual connection. Each subsequent block has convolutions with twice the dilation rate of the previous layer.

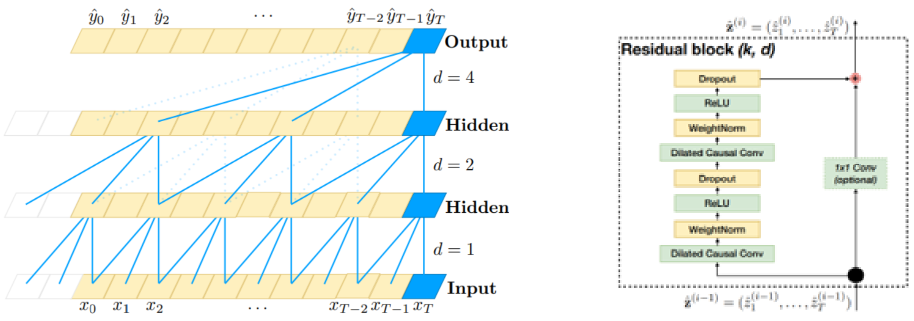

$\hat{y}_0$ $\hat{y}_1$ $\hat{y}_2$ ... $\hat{y}_{T-2}$ $\hat{y}_{T-1}$ $\hat{y}_T$
Output
$d = 4$
Hidden
$d = 2$
Hidden
$d = 1$
Input
x_0 x_1 x_2 ... x_{T-2} x_{T-1} x_T
$\hat{z}^{(i)} = (\hat{z}_1^{(i)}, \dots, \hat{z}_T^{(i)})$
Residual block (k, d)
Dropout
ReLU
WeightNorm
Dilated Causal Conv
Dropout
ReLU
WeightNorm
Dilated Causal Conv
1x1 Conv (optional)
$\hat{z}^{(i-1)} = (\hat{z}_1^{(i-1)}, \dots, \hat{z}_T^{(i-1)})$

Figure 1: *(Left) The expanded receptive field of dilated convolutions where the dilation rate of a layer is twice its the layer preceding it. (Right) Structure of one temporal block with residual connections from Bai et al. [3]. Both images were taken from figures in the paper.*

The key to this architecture's ability to detect long-range interactions in inputs is dilation convolutions that have a greater receptive field compared to their non-dilated counterparts. Similar use of dilations is present in BPNet [9], a deep learning model that utilizes DNA input sequences to predict DNA binding profiles at a base pair resolution, to incorporate signal from up to 1034 base pairs around an output sequence position. The model's successful performance demonstrates the practicality and viability of this type of convolution in a genomics setting.

Model Interpretation and Visualization. The idea of model interpretability originated from Karpathy et al. [10], which is a study that sheds light into what features recurrent models attuned to during training. Gao [11] and Zou et al. [12] provided more insight on how to visualize deep learning models in the context of genomics, such as translating saliency maps into sequence logos which are more typical DNA visualizations in bioinformatics.

In the end, this project explored two types of interpretability techniques. The first is Gradient-weighted Class Activation Mapping (grad-CAM), which extends traditional class activation maps and can be applied to a broader variety of architectures. When fed an input, grad-CAM utilizes the gradients, with respect to a channel, flowing into a final convolutional layer to weight the importance of that channel towards a particular class for the input. The output of this algorithm is a localization map that highlights key regions in an input for a given class, which is crucial in understanding where neural networks focus their attention. While traditionally used in computer vision and images, grad-CAM has been adopted for sequential text data in this project. Hence, the code in Keras' tutorial, "Grad-CAM class activation visualization" [1] [13], was used as a base and modified to process text data and 1-D convolutional layers and produce corresponding visualizations.

The second interpretability method is the dimensionality reduction technique, Uniform Manifold Approximation and Projection (UMAP) [14]. Unlike its more populat alternative t-SNE [15], UMAP can perform non-linear dimension reduction, preserves the global structure of data, produces consistent results across runs, and executes in less time. For these reasons, UMAP was the method chosen to visualize how outputs of layers were clustered in hopes of potentially elucidating the function of those layers. This project uses the publically available package [2] provided by the authors of the original paper in dimension reduction and visualization.

Model Parameter Design and Evaluation. Traditional evaluation classifications methods, like Area Under the Curve (AUC) tend to be applicable for binary cases, not multi-class ones. One solution used in this project is calculating a generalized AUC described in Hand and Till (2001) [16] that can be used to evaluate performance for multi-class classifiers and utilized for this project.

For choosing parameters for models, Gao [11] and Zou et al.[12] provided suitable ranges, such as recommending fewer layers (< 6) and smaller kernel sizes in convolutions, for genomics models. Rio et al. [17] experimented with padding practices, including sparse or dense padding and left, same and right padding, and described their results for a model with genomics application. This initially served as a guide for how inputs were padded for the TCN model, but became less relevant after revising the model's architecture.

Data

NCBI Virus [18] was used to access all publically available GenBank DNA sequences of HCoV-OC43, HCoV-HKU1, HCoV-229E, HCoV-NL63, MERS-CoV, SARS-CoV-1, SARS-CoV-2. This database contains many of the viral genomes from NCBI Nucleotide [19]. The data was in the format of FASTA files, which consist of one line of identifying information (the sample's unique accession id, host species, virus type, GenBank title, and whether the sample was a complete or partial genome). Table 1 highlights relevant statistics about the sequences in the data set and how class labels were defined.

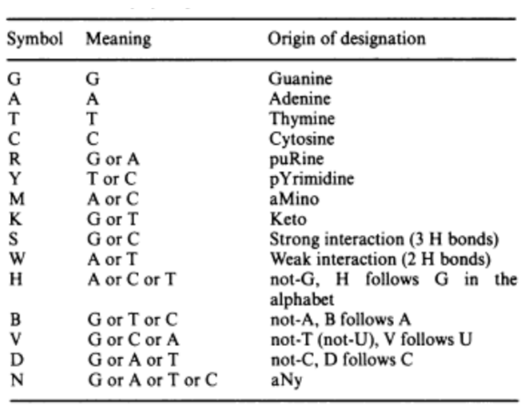
Although there are four nucleotides in nature, the data contains 15 unique letters to present multiple nucleotide choices for a sequence position. These letters are the standard codes provided by International Union of Pure and Applied Chemistry (IUPAC) [20], shown in Figure 2. Thus, the one-hot vectors used to represent sequence positions are of shape 1×15 rather than 1×4.

[1] `https://keras.io/examples/vision/grad_cam/`

[2] `https://umap-learn.readthedocs.io/en/latest/`

Species	Class	# samples	sample length (min, max)
HCoV-229E	0	304	(225, 27307)
HCoV-NL63	1	628	(251, 27832)
HCoV-HKU1	2	399	(143, 29983)
HCoV-OC43	3	1278	(171, 30818)
MERS-CoV	4	738	(147, 30150)
SARS-CoV-1	5	16	(410, 29376)
SARS-CoV-2	6	34606	(64, 30018)

Table 1: *Relevant statistics for the genomics sequences of human coronaviruses from NCBI Virus. While there were many samples from non-human hosts, this project does not focus on classifying them, so those samples were filtered out and not listed above.*



Symbol	Meaning	Origin of designation
G	G	Guanine
A	A	Adenine
T	T	Thymine
C	C	Cytosine
R	G or A	puRine
Y	T or C	pYrimidine
M	A or C	aMino
K	G or T	Keto
S	G or C	Strong interaction (3 H bonds)
W	A or T	Weak interaction (2 H bonds)
H	A or C or T	not-G, H follows G in the alphabet
B	G or T or C	not-A, B follows A
V	G or C or A	not-T (not-U), V follows U
D	G or A or T	not-C, D follows C
N	G or A or T or C	aNy

Figure 2: *IUPAC Codes for Nucleic Acids.*

Approach

Data Preprocessing. All sequences in the FASTA files that were not annotated with 'Homo sapiens' as host species were filtered out, producing the number of examples detailed in Table 1. Most notably, we see there is heavy class imbalance in the data. Initially, to address this issue, the SARS-CoV-1 class was removed and 250 examples were randomly sampled from the remaining classes to create a balanced data set. Sampling, though, may have created artifacts in data for each classes that models might have detected, such as differences in sequence lengths. Therefore, later models were trained on the entire data sets for all species including SARS-CoV-1) except SARS-CoV-2, for which 1500 examples were randomly sampled. This was done to mitigate further class imbalance (SARS-CoV-2 examples outnumber all other classes by a large margin) and because there is less variation in SARS-CoV-2 sequence length which decreases the risk of sampling artifacts.

Next, each base pair of the input sequences were converted into a 1×15 one-hot vector based on the IUPAC codes in Figure 2. Similarly, the output class labels were converted into one-hot vectors of shape 1×7. Then, the data was stratified by class and split into 80/10/10% for training, validation, and testing, respectively. Stratification ensured that each set had a representative number of examples from each class (e.g. 10% of class 0 is in the testing set).

In earlier models, the examples in each data set (i.e. training, validation, and testing) were than converted to RaggedTensors, a type of TensorFlow tensor where elements are non-uniform shapes, to enable batching the data and faster training. In later models, this was removed and examples were fed one-by-one (i.e. batch size = 1) into the model via a generator, which removed the necessity of padding sequences for the TCN model, as TensorFlow's convolutional layer does not support RaggedTensors.

Choosing An Architecture. The first part of the project involved choosing a suitable model architecture. In these models, the Sars-Cov-1 had been replaced with an 'other' class made of randomly generated sequences.

Baseline LSTM Model. The baseline model comprises of 1 LSTM layer followed by a fully-connected layer with softmax activation for classification.

Baseline + Dropout Model. This model adds dropout (rate $= 0.2$) to the inputs and outputs of the LSTM layer (i.e. not to the recurrent units) in the baseline model. Dropout is a regularization technique that can combat overfitting, which is important given the small dataset used for this project.

Stacked LSTM Model. This model consists of 2 LSTM layers, stacked on top of each other, followed by a fully-connected layer with softmax activation. Stacking layers provides more depth to the baseline and enable better classifications. The downside, however, is much greater training times.

Temporal Convolution Network Models. This model uses my implementation of the temporal convolution network that closely follows the one described in Bai [3]. The temporal blocks were constructed as described in Related Works and shown in Figure 1. Each block had a kernel size of 3. The dilation rate of the first block is 1 and increases by a factor os 2 for subsequent layers. The outputs of the temporal blocks are flattened and fed into a fully-connected layer with softmax activation. I tested three models with different number of temporal blocks (3, 4, 5). Dropout (rate $= 0.2$) was applied to the models with 3 and 4 blocks, as they were tested later. Because TCNs do not rely on recurrences, they are much faster to train.

Revision and Finalization. TCNs, with the same type of increase in dilation rate between subsequent blocks, were chosen as the underlying architecture as they outperformed all LSTM models and took less time to train. Concerned about artifacts in the data sets, such as differences in the average and spread of sequence lengths among classes, I then modified the TCN architectures to allow for inputs of varying lengths. Specifically, examples were not batched and outputs of the temporal blocks underwent a global average pooling before the fully connected layer with softmax activation. Global average pooling acts over the temporal/sequential dimension and averages the information from each sequence position. This is required to create uniformly shaped outputs that can be fed into the subsequent fully connected layer. The final architecture is shown below in Figure 3.

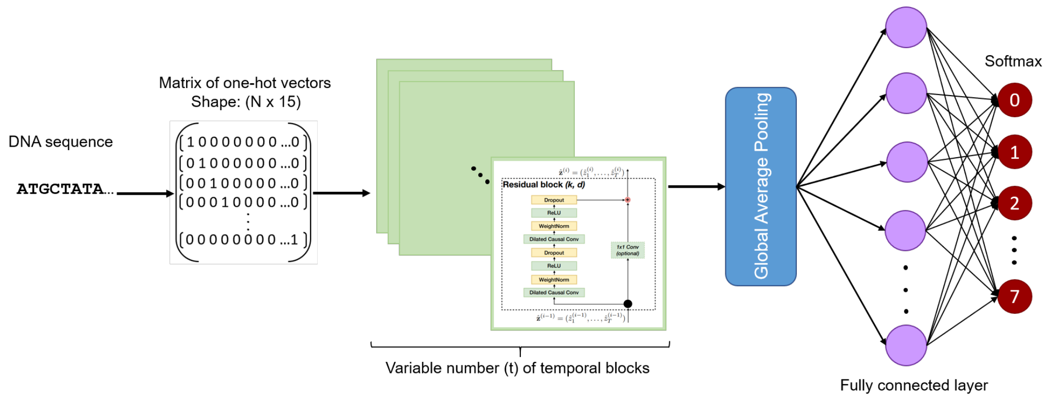


Figure 3: *Depiction of workflow. Viral DNA sequences of variable lengths, N, and with 'Homo sapien' hosts are converted a one-hot vector representation. This is fed into the model composed of a variable number of consecutive temporal blocks with residual connections (i.e. despite the diagram, the temporal blocks are not on the same level, but rather have a sequential relationship), a global average pooling layer over the sequential dimension to standardize shapes, a fully connected layer, and a softmax classifer. The class with the highest probability outputted by the softmax activation determines the predicted class of the input.*

Weight Loss Function. During training, the individual example losses were weighted based on their actual class label, as listed below:

$$class\ 0\!: 4.0,\ class\ 1\!: 2.0,\ class\ 2\!: 3.0,\ class\ 3\!: 1.0,\ class\ 4\!: 4.0,\ class\ 5\!: 100.0,\ class\ 6\!: 2.0$$

These weights were used to incentivize correct classification of under-represented classes. They were chosen by splitting the classes into viruses with more and less severe disease prognosis, dividing the number of examples by the max number in each group (1500 and 1140 respectively), multiplying the more dangerous virus group by 2, rounding, and capping at 100. This scheme adjusts for class imbalance and differences in the importance of correct classification among classes. Nevertheless, their values are still considered arbitrarily chosen, so exploring different weighting schemes is done during experimentation.

Performance Evaluation and Metrics. Different types of multi-class AUC and ROC curves were originally used to gauge and visualize model performance for the artificially balanced data set and models prior to the milestone. Later, F1-score and Precision-Recall curves replaced these metrics since the final models were trained on an imbalanced data set. Both macro and micro F1-Score was measured. The former implicitly favors correct classification of minority classes while the latter implicitly favors more correct classification of classes with many examples. Therefore, discrepancies between the metrics offer insight on which classes models struggle with, so both are used to gauge performance. Ideally, though, the two F1-Scores should be similar. Comparisons of model performance, especially when determining the best model during validation, will be done manually and use both metrics; that is, models with low micro F1-Scores will not be the best even if they have the highest macro F1-Scores and vice versa.

Model Interpretation and Visualization. Grad-CAM was used on (generally) randomly chosen sequences, including those which were correctly and incorrectly classified, but in the same class. The vectors outputs were plotted against sequence position and analyzed for patterns in peaks, in which the relative magnitude of these peaks indicate how attuned the models were to a position. Additionally, for all layers in the model,

Experiments

Model	C. Acc.	AUC (mi, ma, mlt)	F1-Score (mi, ma)	T. Time (hrs)
Baseline	0.48	(0.86, 0.84, 0.83)	(0.48, 0.42)	3
Base + Dropout	0.40	(0.81, 0.79, 0.77)	(0.40, 0.29)	2
Stacked	0.51	(0.89, 0.87, 0.86)	(0.51, 0.42)	12
TCN-3 + Dropout	0.93	(0.99, 0.99, 0.99)	(0.93, 0.92)	3
TCN-4 + Dropout	0.73	(0.94, 0.98, 0.98)	(0.73, 0.75)	2
TCN-5	0.45	(0.82, 0.93, 0.92)	(0.45, 0.33)	2.5

Table 2: *Performance statistics on the validation set for models tested while choosing an architecture. 'C. Acc.' stands for Categorical Accuracy, 'mi' stands for micro, 'ma' stands for macro, 'mlt' AUC is the multi-class AUC described in Hand and Till [16], and 'T. Time' is model Training Time.*

Table 2 displays statistics for models tested while choosing an architecture (i.e. up to milestone). Despite the same training hyperparameters, LSTM-based models performed worse during test time and took more time to train. As a result, subsequent experiments focused on exploring TCN-based models.

Optimizing Parameters in Architecture Design. Convolution kernel size, initial dilation rate, and number of temporal blocks all influence the maximum receptive field achieved by the final convolutional layer, which is important if there are key relationships necessary for classificaiton between nucleotides spaced farther away in viral genomes. Thus, the first set of experiments after finalizing the model's architecture was to adjust some of these parameters as well as the number of channels in the convolution. Combinations of parameters tested are denoted in Table 3.

Model	temporal blocks	kernel size	channels
1	4	3	64
2	6	3	64
3	8	3	64
4	4	5	64
5	4	7	64
6	4	3	128
7	4	3	256

Table 3: *Combinations of model parameters tested and trained using Adam with a learning rate of* $4e-5$*, dropout rate of* 0.2*, patience for* 2 *epochs before early stopping, and weighted loss function.*

Initial dilation rates greater than one prevents the model from including immediately adjacent nucleotides in one convolution. This is suboptimal in an biological context as every three nucleotides defines one amino acids, which are important in determining the structure and function of proteins encoded by these DNA sequences. Therefore, the initial dilation rate remains 1. The results from this experiment are shown in Figure 4. **The best set of model parameters was determined to be 4 temporal blocks that included convolutions with 7$\times$7 kernels and 64 channels.**

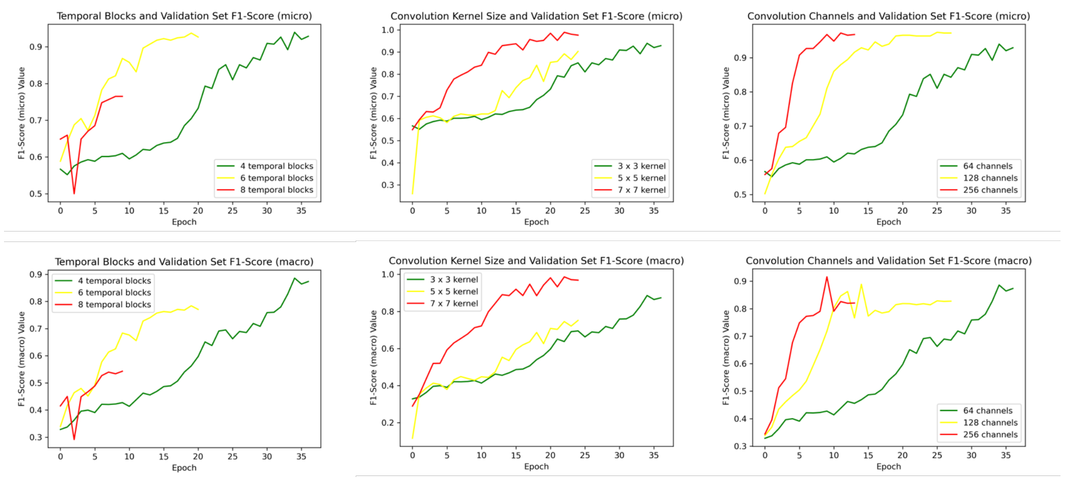


Figure 4: *Validation set performance for models with different model parameters.*

Hyperparameter Search. This experiment focused on tuning hyperparameters involved in training of the best model from the previous experiment. Because optimizing with Adam attained decent results and has many theoretical advantages, alternative optimizers were not explored. Rather, as shown in Table 4, various values of Adam's learning rate and hyperparameters used in regularization methods, namely patience in early stopping and the probability of an input element being drop during Dropout within temporal blocks, were tested. This experiment was important for choosing suitable training conditions that promote the model to learn optimal weights while minimizing the chance of overfitting. The results from this experiment are shown in Figure 5. **The best set of training hyperparameters was determined to be a learning rate of 4e-5, a dropout rate of 0.1, and patience for 2 epochs.**

Model	learning rate	dropout probability	patience (epochs)
7	4e-5	0.2	2
8	5e-4	0.2	2
9	1e-5	0.2	2
10	4e-5	0.1	2
11	4e-5	0.0	2
12	4e-5	0.2	4

Table 4: *Combination of training hyperparameters tested on the best model with 4 temporal blocks and 7 $\times$ 7 kernels and 64 channels in convolutional layers.*

Best Model Evaluation. After determining the best model parameter and training hyperparameters among the combinations tested, the test set was evaluated using this model, and results are shown below:

Categorical Accuracy	F1-Score (micro)	F1-Score (macro)
0.9935	0.9935	0.9917

These results suggest that the model has achieved good performance, even in minority classes. This fact is also supported by analyzing the confusion matrix, shown in Figure 6, for example, when the model is able to correctly classify the SARS-CoV-1 group, likely as a result of the heavy

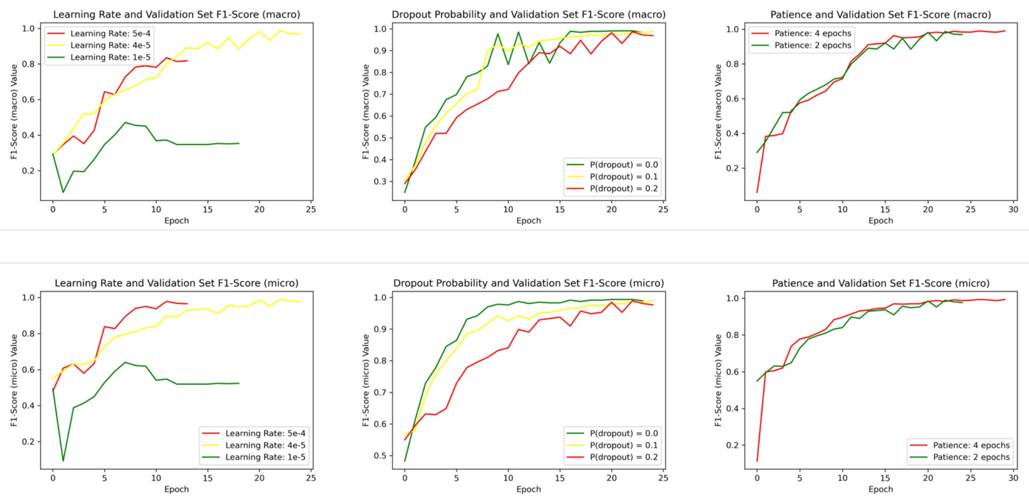


Figure 5: *Validation set performance for models with different model parameters.*

weighting it had in the loss function. The confusion matrix also reveals that the model had the most misclassifications (2) for MERS-CoV and that there is no apparent pattern in misclassifications. Similarly, the precision-recall curves in Figure 6 shows that the model requires little trade off between having a high sensitivity and a high positive predictive power as it has both.

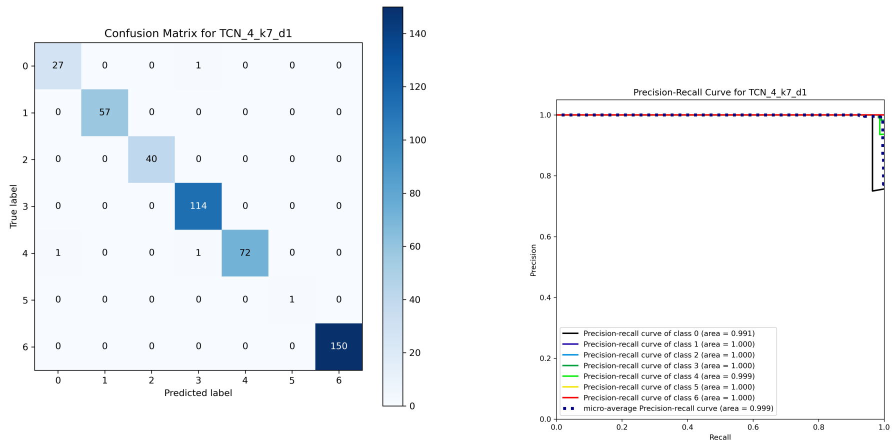


Figure 6: *Confusion matrix and precision recall curve generated from evaluation on the test set*

Class Attenuation Using grad-CAM. Henceforth, all interpretability experiments only involve the best model trained under the best training conditions. While the goal of this experiment was to better understand which input sequence positions are more attuned to for a given class, it was difficult to analyze the results of this experiment in practice. Reasons included long sequence lengths and differences in lengths. The former made it impossible to fit all positions in a single plot and analyze together, especially for complete genomes that had around 29000 positions (for 29000 base pairs). These challenges made searching for peak patterns very time intensive, and along with time constraints, I was not able to analyze enough input sequences to find significant peak patterns in input sequences that could indicate unique motifs for certain classes.

Supervised Dimension Reduction of Layer Outputs Using UMAP. To run a UMAP analysis, the outputs of all temporal block were first averaged pooled globally, separately, in order to standardize shapes between input sequences. This step was skip for the outputs of the global average pooling layer after the temporal blocks. Then, the UMAP algorithm, with 15 nearest neighbors; a minimum distance of 0.1 between embedding points; and the euclidean distance as parameters, reduced these outputs into 2 dimensions. The results are shown in Figure 7. Despite the algorithm knowing each input's class label, the results show that clusters are not based on classes. One area of further exploration is to elucidate features that define these clusters.

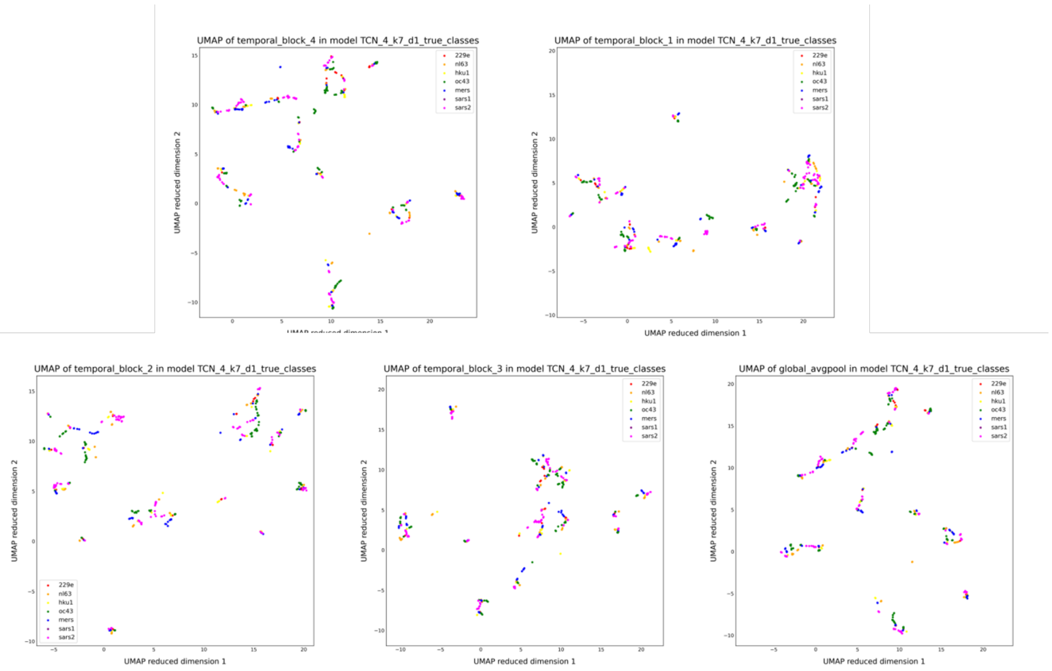


Figure 7: *UMAP plots for embeddings representations of layer outputs. Different colors represent different classes.*

Conclusions

This project shows that temporal convolution networks can be trained to accurately classify human coronaviruses and is a better choice than recurrent methods, such as LSTMs, because of better model performance and shorter training times. Additionally, it seems that smaller models obtain better performance since the best model had the smallest number of convolution channels and number of temporal blocks. However, a larger kernel size did improve performance, which may indicate that local features contribute more to accurate classification than relationships between nucleotides farther away from each other. With the addition of a weighted loss function, the model was able to achieve good performance on an imbalanced data set, as evidenced by a macro and micro F1-Score of 0.99.

Unfortunately, interpretability techniques tested did not shed light about how parts of the model operated. Therefore, avenues for future work included further exploration of interpretability techniques or refining ones currently explored. Additionally, given the small sample size of classes, especially SARS-CoV-1, it would be useful to test if the model still performs well on a larger data set.

References

[1] Anthony R. Fehr and Stanley Perlman. Coronaviruses: An overview of their replication and pathogenesis. *Methods in Molecular Biology*, 2015.

[2] Coronavirus Resource Center Johns Hopkins University. Mortality analyses. `https://coronavirus.jhu.edu/data/mortality`.

[3] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018.

[4] Peter T. Habib, Alsamman M. Alsamman, M. Saber-Ayad, S. Hassanein, and A. Hamwieh. Covidier: A deep-learning tool for coronaviruses genome and virulence proteins classification. *bioRxiv*, 2020.

[5] Alejandro Lopez-Rincon, Alberto Tonda, Lucero Mendoza-Maldonado, Eric Claassen, Johan Garssen, and Aletta D. Kraneveld. Accurate identification of sars-cov-2 from viral genome sequences using deep learning. *bioRxiv*, 2020.

[6] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models, 2015.

[7] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification, 2016.

[8] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai kin Wong, and Wang chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting, 2015.

[9] Žiga Avsec, Melanie Weilert, Avanti Shrikumar, Amr Alexandari, Sabrina Krueger, Khyati Dalal, Robin Fropf, Charles McAnany, Julien Gagneur, Anshul Kundaje, and Julia Zeitlinger. Deep learning at base-resolution reveals motif syntax of the cis-regulatory code. *BioRxiv*, 2019.

[10] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks, 2015.

[11] Xin Gao. Deep learning methods for mining genomic sequence patterns, 2018.

[12] James Zou, Mikael Huss, Abubakar Abid, Pejman Mohammadi, Ali Torkamani, and Amalio Telenti. A primer on deep learning in genomics. *Nature Genetics*, 2018.

[13] fchollet. Grad-cam class activation visualization. 2020. `https://keras.io/examples/vision/grad_cam/`.

[14] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2018.

[15] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne, 2008.

[16] David J. Hand and Robert J. Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning*, 2001.

[17] Angela Lopez del Rio, Maria Martin, Alexandre Perera-Lluna, and Rabie Saidi. Effect of sequence padding on the performance of deep learning models in archaeal protein functional prediction. *Nature*, 2020.

[18] Eneida L Hatcher, Sergey A Zhdanov, Yiming Bao, Alga Blinkova, Eric P Nawrocki, Yuri Ostapchuck, Alejandro A Schäffer, and J Rodney Brister. Virus variation resource - improved response to emergent viral outbreaks. *Nucleic Acids Research*, 2016.

[19] National Center for Biotechnology Information (NCBI). Nucleotide database. `https://www.ncbi.nlm.nih.gov/nucleotide/`.

[20] Nomenclature for incompletely specified bases in nucleic acid sequences. recommendations 1984. nomenclature committee of the international union of biochemistry (nc-iub). *Proceedings of the National Academy of Sciences of the United States of America*, 1986.